Statistical Model Checking for Networks of Priced Timed Automata *

Alexandre David[1], Kim G. Larsen[1], Axel Legay[2], Marius Mikučionis[1], Danny Bøgsted Poulsen[1], Jonas van Vliet[1], and Zheng Wang[3]

[1] Aalborg University, Denmark
[2] INRIA/IRISA Rennes, France
[3] East China Normal University, China

Abstract This paper offers a natural stochastic semantics of Networks of Priced Timed Automata (NPTA) based on races between components. The semantics provides the basis for satisfaction of Probabilistic Weighted CTL properties (PWCTL), conservatively extending the classical satisfaction of timed automata with respect to TCTL. In particular the extension allows for hard real-time properties of timed automata expressible in TCTL to be refined by performance properties, e.g. in terms of probabilistic guarantees of time- and cost-bounded properties. A second contribution of the paper is the application of Statistical Model Checking (SMC) to efficiently estimate the correctness of non-nested PWCTL model checking problems with a desired level of confidence, based on a number of independent runs of the NPTA. In addition to applying classical SMC algorithms, we also offer an extension that allows to efficiently compare performance properties of NPTAs in a parametric setting. The third contribution is an efficient tool implementation of our result and applications to several case studies.

1 Introduction

Model Checking (MC) [11] is a widely recognised approach to guarantee the correctness of a system by checking that any of its behaviors is a model for a given property. There are several variants and extensions of MC aiming at handling real-time and hybrid systems with quantitative constraints on time, energy or more general continuous aspects [1–3,6]. Within the field of embedded systems these formalisms and their supporting tools [16, 29, 30, 32] are now successfully applied to time- and energy-optimal scheduling, WCET analysis and schedulability analysis.

Compared with traditional approaches, a strong point of real-time model checking is that it (in principle) only requires a model to be applicable, thus extensions to multi-processor setting is easy. A weak point of model checking is the state-space explosion, i.e. the exponential growth in the analysis effort measured in the number of model-components. Another limitation of real-time

* Work partially supported by VKR Centre of Excellence – MT-LAB and by an "Action de Recherche Collaborative" ARC (TP)I.

model checking is that it merely provides – admittedly most important – hard quantitative guarantees, e.g. the worst case response time of a recurrent task under a certain scheduling principle, the worst case execution time of a piece of code running on a particular execution platform, or the worst case time before consensus is reached by a real-time network protocol. In addition to these hard guarantees, it would be desirable in several situations to obtain refined performance information concerning likely or expected behaviors in terms of timing and resource consumption. In particular, this would allow to distinguish and select between systems that perform identically from a worst-case perspective.

As a first contribution we propose a stochastic semantics for Priced Timed Automata (PTA), whose clocks can evolve with different rates, while[4] being used with no restrictions in guards and invariants. Networks of PTAs (NPTA) are created by composing PTAs via input and output actions. More precisely, we define a natural stochastic semantics for networks of NPTAs based on races between components being composed. We shall observe that such race can generate arbitrarily complex stochastic behaviors from simple assumptions on individual components. We shall see that our semantics cannot be emulated by applying the existing stochastic semantic of [4, 8] to the product of components. Other related work includes the very rich framework of stochastic timed systems of MoDeST [10]. Here, however, general hybrid variables are not considered and parallel composition does not yield fully stochastic models. For the notion of probabilistic hybrid systems considered in [31] the choice of time is resolved non-deterministically rather than stochastically as in our case. Moreover, based on the stochastic semantics, we are able to express refined performance properties, e.g. in terms of probabilistic guarantees of time- and cost-bounded properties[5].

To allow for the efficient analysis of probabilistic performance properties we propose to work with Statistical Model Checking (SMC) [28, 35], an approach that has been proposed as an alternative to avoid an exhaustive exploration of the state-space of the model. The core idea of SMC is to monitor some simulations of the system, and then use results from the statistic area (including sequential hypothesis testing or Monte Carlo simulation) in order to decide whether the system satisfies the property with some degree of confidence.

Thus, as a second contribution, we provide an efficient implementation of several existing SMC algorithms that we use for checking the correctness of NPTAs with respect to a stochastic extension of cost-constrained temporal logic – this extension being conservative with respect to the classical (non-stochastic) interpretation of the logic. We shall observe that two timed bisimilar NPTAs may be distinguisable by PWCTL. The series of algorithms we implemented includes the sequential hypothesis test by Wald [34] as well as a quantitative approach [18]. Our implementation relies on a new efficient algorithm for generating runs of NPTAs in a random manner. In addition, we also propose another SMC algorithm to compare the probabilities of two properties without computing them individually – which is useful to compare the performances of a program

[4] in contrast to the usual restriction of priced timed automata [3, 6]

[5] Clocks with different rates can be used to model costs.

with one of its evolutions at cheap cost. This probability comparison problem, which is far beyond the scope of existing time model checking approaches, can be approximated with an extension of the sequential hypothesis testing and has the advantage of unifying the confidence in the comparison. In addition to be the first to apply such extension in the context of formal verification, we also propose a new variant that allows to reuse existing results in parallel when comparing the properties on different timed bounds.

Finally, one of the most interesting contribution of our work takes the form of a series of new case studies that are analyzed with a new stochastic extension of Uppaal [13]. Particularly, we show how our approach can be used to resolve scheduling problems. Such problems are defined using Duration Probabilistic Automata (DPA) [24], a new and natural model for specifying list of tasks and shared resources. We observe that our approach is not only more general, but also an order of magnitude faster than the hypothesis testing engine recently implemented in the Prism toolset. Our work thus presents significant advances in both the modeling and the efficient verification of network of complex systems.

Related work. Some works on probabilistic semantics of timed automata have already been discussed above. Simulation-based approaches such as Monte Carlo have been in use since decades, however the use of simulation and hypothesis testing to reason on formal models is a more recent advance. First attempts to apply hypothesis testing on stochastic extension of Hennessy-Milner logic can be found in [23]. In [35, 37], Younes was the first to apply hypothesis testing to stochastic systems whose properties are specified with (bounded) temporal logic. His approach is implemented in the Ymer toolset [36] and can be applied on time-homogeneous generalized semi-Markov processes, while our semantics addresses the composition of stochastic systems allowing to compose a global system from components and reason about communication between independent processes. In addition to Younes work we explore continuous-time features, formalize and implement Wald's ideas where the probability comparison can be evaluated on NPTA processes. In a recent work [38], Zuliani et al. extended the SMC approach to hybrid systems. Their work is a combination of [20] and [12] based on Simulink models (non-linear hybrid systems), whereas our method is specialised to networks of priced timed automata where model-checking techniques can be directly applicable using the same tool suite. In addition we provide means of comparing performances without considering individual probabilities. Finally, a very recent work [9] proposes partial order reduction techniques to resolve non-determinism between components rather than defining a unique stochastic distribution on their product behaviors. While this work is of clear interest, we point out that the application of partial order may considerably increase the computation time and for some models partial orders cannot resolve non-determinism, especially when considering continuous time [25]. Finally, we mention [22] that proposes a stochastic semantics to Uppaal's models through simulation. This work does not consider race between components and offers no tool implementation.

2 Network of Priced Timed Automata

We consider the notion of *Networks of Priced Timed Automata (NPTA)*, generalizing that of regular timed automata (TA) in that clocks may have different rates in different locations. In fact, the expressive power (up to timed bisimilarity) of NPTA equals that of general linear hybrid automata (LHA) [1], rendering most problems – including that of reachability – undecidable.

Let X be a finite set of variables, called *clocks*[6]. A *clock valuation* over X is a mapping $\nu : X \to \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is the set of nonnegative reals. We write $\mathbb{R}^X_{\geq 0}$ for the set of clock valuations over X. Let $r : X \to \mathbb{N}$ be a *rate vector*, assigning to each clock of X a rate. Then, for $\nu \in \mathbb{R}^X_{\geq 0}$ and $d \in \mathbb{R}_{\geq 0}$ a delay, we write $\nu + r \cdot d$ for the clock valuation defined by $(\nu + r \cdot d)(x) = \nu(x) + r(x) \cdot d$ for any clock $x \in X$. We denote by $\mathbb{N}^X$ the set of all rate vectors. If $Y \subseteq X$, the valuation $\nu[Y]$ is the valuation assigning 0 when $x \in Y$ and $\nu(x)$ when $x \notin Y$. An *upper bounded (lower bound) guard* over X is a finite conjunction of simple clock bounds of the form $x \sim n$ where $x \in X$, $n \in \mathbb{N}$, and $\sim \in \{<, \leq\}$ ($\sim \in \{>, \geq\}$) We denote by $\mathcal{U}(X)$ ($\mathcal{L}(X)$ the set of upper (lower) bound guards over X, and write $\nu \models g$ whenever ν is a clock valuation satisfying the guard g. Let $\Sigma = \Sigma_i \uplus \Sigma_o$ be a disjoint sets of input and output actions.

Definition 1. *A* Priced Timed Automaton *(PTA) is a tuple* $\mathcal{A} = (L, \ell_0, X, \Sigma, E, R, I)$ *where: (i)* L *is a finite set of locations, (ii)* $\ell_0 \in L$ *is the initial location, (iii)* X *is a finite set of clocks, (iv)* $\Sigma = \Sigma_i \uplus \Sigma_o$ *is a finite set of actions partitioned into inputs* (Σ_i) *and outputs* (Σ_o)*, (v)* $E \subseteq L \times \mathcal{L}(X) \times \Sigma \times 2^X \times L$ *is a finite set of edges, (vi)* $R : L \to \mathbb{N}^X$ *assigns a rate vector to each location, and (viii)* $I : L \to \mathcal{U}(X)$ *assigns an invariant to each location.*

The semantics of NPTAs is a timed labelled transition system whose states are pairs $(\ell, \nu) \in L \times \mathbb{R}^X_{\geq 0}$ with $\nu \models I(\ell)$, and whose transitions are either delay $(\ell, \nu) \xrightarrow{d} (\ell, \nu')$ with $d \in \mathbb{R}_{\geq 0}$ and $\nu' = \nu + R(\ell) \cdot d$, or discrete $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ if there is an edge (ℓ, g, a, Y, ℓ') such that $\nu \models g$ and $\nu' = \nu[Y]$. We write $(\ell, \nu) \rightsquigarrow (\ell', \nu')$ if there is a finite sequence of delay and discrete transitions from (ℓ, ν) to (ℓ', ν').

Networks of Priced Timed Automata Following the compositional specification theory for timed systems in [14], we shall assume that NPTAs are: (1)[*Input-enabled*:] for all states (ℓ, ν) and input actions $\iota \in \Sigma_i$, for all TAs j, there is an edge $(\ell^j, g, \iota, Y, \ell^{j'})$ such that $\nu \models g$, (2) [*Deterministic*:] for all states (ℓ, ν) and actions $a \in \Sigma$, whenever $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ and $(\ell, \nu) \xrightarrow{a} (\ell'', \nu'')$ then $\ell' = \ell''$ and $\nu' = \nu''$, and (3) [*Non-zenos*:] time always diverge. Moreover, different automata synchronize on matching inputs and outputs as a standard broadcast synchronization [17].

Whenever $\mathcal{A}^j = (L^j, X^j, \Sigma^j, E^j, R^j, I^j)$ ($j = 1 \ldots n$) are NPTA, they are *composable* into a *closed network* iff their clock sets are disjoint ($X^j \cap X^k = \emptyset$

[6] We will (mis)use the term "clock" from timed automata, though in the setting of NPTAs the variables in X are really general real-valued variables.

when $j \neq k$), they have the same action set ($\Sigma = \Sigma^j = \Sigma^k$ for all j,k), and their output action-sets provide a partition of Σ ($\Sigma_o^j \cap \Sigma_o^k = \emptyset$ for $j \neq k$, and $\Sigma = \cup_j \Sigma_o^j$). For $a \in \Sigma$ we denote by $c(a)$ the unique j with $a \in \Sigma^j$.

Definition 2. *Let $\mathcal{A}^j = (L^j, X^j, \Sigma, E^j, R^j, I^j)$ (with $j = 1 \ldots n$) be composable NPTAs. Their* composition $(\mathcal{A}_1 \,|\, \ldots \,|\, \mathcal{A}_n)$ *is the NPTA $\boldsymbol{\mathcal{A}} = (L, X, \Sigma, E, R, L)$ where (i) $L = \times_j L^j$, (ii) $X = \cup_j X^j$, (iii) $R(\boldsymbol{\ell})(x) = R^j(\ell^j)(x)$ when $x \in X^j$, (iv) $I(\boldsymbol{\ell}) = \cap_j I(\ell^j)$, and (v) $(\boldsymbol{\ell}, \cap_j g_j, a, \cup_j r_j, \boldsymbol{\ell'}) \in E$ whenever $(\ell_j, g_j, a, r_j, \ell'_j) \in E^j$ for $j = 1 \ldots n$.*

Example 1. Let A, B, T and AB be the priced timed automata depicted in Fig. 1 [7] Then A, B and T are composable as well as AB and T. In fact the composite systems $(A|B|T)$ and $(AB|T)$ are timed (and priced) bisimilar, both having the transition sequence:

$$\big((A_0, B_o, T_0), [x=0, y=0, C=0]\big) \xrightarrow{1} \xrightarrow{a!} \big((A_1, B_0, T_1), [x=1, y=1, C=4]\big) \xrightarrow{1} \xrightarrow{b!} \big((A_1, B_1, T_2), [x=2, y=2, C=6]\big),$$

demonstrating that the final location T_3 of T is reachable with cost 6.

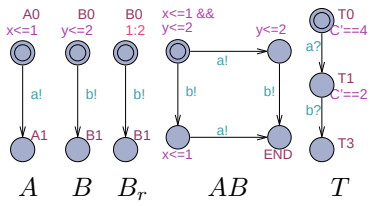

Figure 1: Three composable NPTAs: A, B and T; A, B_r and T; and AB and T.

3 Probabilistic Semantics of NPTA

Continuing Example 1 we may realise that location T_3 of the component T is reachable within cost 0 to 6 and within total time 0 and 2 in both $(A|B|T)$ and $(AB|T)$ depending on when (and in which order) A and B (AB) chooses to perform the output actions $a!$ and $b!$. Assuming that the choice of these time-delays is governed by probability distributions, we will in this section define a probability measure over sets of infinite runs of networks of NPTAs.

In contrast to the probabilistic semantics of timed automata in [4, 8] our semantics deals with networks and thus with races between components. Let $\mathcal{A}^j = (L^j, X^j, \Sigma, E^j, R^j, I^j)$ ($j = 1 \ldots n$) be a collection of composable NPTAs. Under the assumption of input-enabledness, disjointness of clock sets and output actions, states of the the composite NPTA $\boldsymbol{\mathcal{A}} = (\mathcal{A}_1 \,|\, \ldots \,|\, \mathcal{A}_n)$ may be seen as tuples $\mathbf{s} = (s_1, \ldots, s_n)$ where s_j is a state of $\mathcal{A}^j$, i.e. of the form (ℓ, ν) where $\ell \in L^j$ and $\nu \in \mathbb{R}_{\geq 0}^{X^j}$. Our probabilistic semantics is based on the principle of independency between components. Repeatedly each component decides on its own – based on a given delay density function and output probability function – how much to delay before outputting and what output to broadcast at that moment. Obviously, in such a race between components the outcome will be determined by the component that has chosen to output after the minimum delay: the output is broadcast and all other components may consequently change state.

[7] The broadcast synchronization we use allows us to ignore missing input transitions that may otherwise be added as looping transitions.

Probabilistic Semantics of NPTA Components Let us first consider a component $\mathcal{A}^j$ and let St^j denote the corresponding set of states. For each state $s = (\ell, \nu)$ of $\mathcal{A}^j$ we shall provide probability distributions for both delays and outputs. In this presentation, we restrict to uniform and universal distributions, but arbitrary distributions can be considered.

The *delay density function* μ_s over delays in $\mathbb{R}_{\geq 0}$ will be either a uniform or an exponential distribution depending on the invariant of ℓ. Denote by E_ℓ the disjunction of guards g such that $(\ell, g, o, -, -) \in E^j$ for some output o. Denote by $d(\ell, \nu)$ the infimum delay before enabling an output, i.e. $d(\ell, \nu) = \inf\{d \in \mathbb{R}_{\geq 0} : \nu + R^j \cdot d \models E_\ell\}$, and denote by $D(\ell, \nu)$ the supremum delay, i.e. $D(\ell, \nu) = \sup\{d \in \mathbb{R}_{\geq 0} : \nu + R^j \cdot d \models I^j(\ell)\}$. If $D(\ell, \nu) < \infty$ then the delay density function μ_s is a uniform distribution on $[d(\ell, \nu), D(\ell, \nu)]$. Otherwise – that is $I^j(\ell)$ does not put an upper bound on the possible delays out of s – the delay density function μ_s is an exponential distribution with a rate $P(\ell)$, where $P : L^j \to \mathbb{R}_{\geq 0}$ is an *additional* distribution rate component added to the NPTA $\mathcal{A}^j$. For every state $s = (\ell, \nu)$, the *output probability function* γ_s over Σ_o^j is the uniform distribution over the set $\{o : (\ell, g, o, -, -) \in E^j \wedge \nu \models g\}$ whenever this set is non-empty [8]. We denote by s^o the state after the output of o. Similarly, for every state s and any input action ι, we denote by s^ι the state after having received the input ι.

Probabilistic Semantics of Networks of NPTA We shall now see that while the stochastic semantics of each PTA is rather simple (but quite realistic), arbitrarily complex stochastic behavior can be obtained by their composition.

Reconsider the closed network $\boldsymbol{\mathcal{A}} = (\mathcal{A}_1 \,|\, \ldots \,|\, \mathcal{A}_n)$ with a state space $\mathsf{St} = \mathsf{St}_1 \times \cdots \times \mathsf{St}_n$. For $\mathbf{s} = (s_1, \ldots, s_n) \in \mathsf{St}$ and $a_1 a_2 \ldots a_k \in \Sigma^*$ we denote by $\pi(\mathbf{s}, a_1 a_2 \ldots a_k)$ the set of all maximal runs from $\mathbf{s}$ with a prefix $t_1 a_1 t_2 a_2 \ldots t_k a_k$ for some $t_1, \ldots, t_n \in \mathbb{R}_{\geq 0}$, that is runs where the i'th action a_i has been outputted by the component $A_{c(a_i)}$. We now inductively define the following measure for such sets of runs:

$$\mathbb{P}_{\boldsymbol{\mathcal{A}}}\big(\pi(\mathbf{s}, a_1 \ldots a_n)\big) = \int_{t \geq 0} \mu_{s_c}(t) \cdot \Big(\prod_{j \neq c} \int_{\tau > t} \mu_{s_j}(\tau) d\tau \Big) \cdot \gamma_{s_c{}^t}(a_1) \cdot \mathbb{P}_{\boldsymbol{\mathcal{A}}}\big(\pi(\mathbf{s}^t)^{a_1}, a_2 \ldots a_n)\big) dt$$

where $c = c(a_1)$, and as base case we take $P_{\boldsymbol{\mathcal{A}}}(\pi(\mathbf{s}), \varepsilon) = 1$.

This definition requires a few words of explanation: at the outermost level we integrate over all possible initial delays t. For a given delay t, the outputting component $c = c(a_1)$ will choose to make the broadcast at time t with the stated density. Independently, the other components will choose to a delay amount, which – in order for c to be the winner – must be larger than t; hence the product of the probabilities that they each make such a choice. Having decided for making the broadcast at time t, the probability of actually outputting a_1 is included. Finally, in the global state resulting from all components having delayed t time-units and changed state according to the broadcasted action a_1 the probability of runs according to the remaining actions $a_2 \ldots a_n$ is taken into account.

[8] otherwise a specific weight distribution can be specified and used instead.

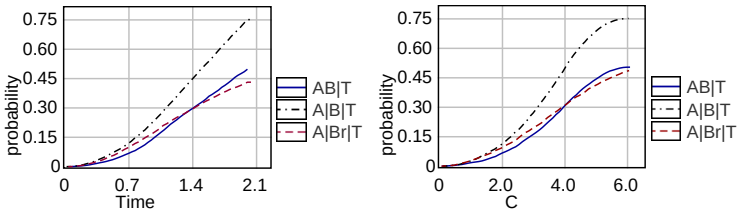

Figure 2: Cumulative probabilities for time and cost-bounded reachability of T_3.

Logical Properties Following [26], the measure $\mathbb{P}_{\mathcal{A}}$ may be extended in a standard and unique way to the σ-algebra generated by the sets of runs (so-called cylinders) $\pi(\mathbf{s}, a_1 a_2 \dots a_n)$. As we shall see this will allow us to give proper semantics to a range of probabilistic time- and cost-constrained temporal properties. Let $\mathcal{A}$ be a NPTA. Then we consider the following non-nested PWCTL properties:

$$\psi \ ::= \ \mathbb{P}\big(\Diamond_{C\leq c}\varphi\big) \sim p \ \mid \ \mathbb{P}\big(\Box_{C\leq c}\varphi\big) \sim p$$

where C is an observer clock (of $\mathcal{A}$), φ a state-property (wrt. $\mathcal{A}$) , $\sim \in \{<, \leq, =, \geq, >\}$, and $p \in [0,1]$. This logic is a stochastic extension of the classical WCTL logic for non-stochastic systems, where the existential quantifier is replaced by a probability operator. For the semantics let $\mathcal{A}^*$ be the modification of $\mathcal{A}$, where the guard $C \leq c$ has been conjoined to the invariant of all locations and an edge $(\ell, \varphi, o_\varphi, \emptyset, \ell)$ has been added to all locations ℓ, where o_φ is a new output action. Then:

$$\mathcal{A} \models \mathbb{P}\big(\Diamond_{C\leq c}\varphi\big) \sim p \ \text{ iff } \ \mathbb{P}_{\mathcal{A}^*}\Big(\bigcup_{\sigma\in\Sigma^*} \pi(s_0, \sigma o_\varphi)\Big) \sim p$$

which is well-defined since the σ-algebra on which $\mathbb{P}_{\mathcal{A}^*}$ is defined is closed under countable unions and finite intersections. To complete the semantics, we note that $\mathbb{P}(\Box_{C\leq c}\varphi) \sim p$ is equivalent to $(1-p) \sim \mathbb{P}(\Diamond_{C\leq c}\neg\varphi)$. [9]

Compared with previous stochastic semantics of timed automata (see e.g., [4,8]), we emphasize the novelty of the semantics of NPTA in terms of RACES between components, truthfully reflecting their independencies. In particular our stochastic semantics of a network $(A_1|..|A_n)$ is significantly different from that obtained by applying the stochastic semantics of [4,8] to a product construction $A_1 A_2 \dots A_n$, as information about independencies are lost. So though $(A_1|..|A_n)$ and $A_1 A_2 \dots A_n$ are timed bisimilar they are in general not probabistic timed bisimilar, and hence distinguishable by PWCTL. The situation is illustrated with the following example.

Example 2. Reconsider the Example of Fig. 1. Then it can be shown that $(A|B|T) \models \mathbb{P}\big(\Diamond_{t\leq 2} T_3\big) = 0.75$ and $(A|B|T) \models \mathbb{P}\big(\Diamond_{C\leq 6} T_3\big) = 0.75$, whereas

[9] We also note that the above (stochastic) interpretation of PWCTL is a conservative extension of the classical (non-stochastic) interpretation of WCTL, in the sense that $\mathcal{A} \models \mathbb{P}\big(\Diamond_{C\leq c}\varphi\big) > 0$ implies $\mathcal{A}_n \models \mathsf{E}\Diamond_{C\leq c}\varphi$, where $\mathcal{A}_n$ refers to the standard non-stochastic semantics of $\mathcal{A}$.

$(AB|T) \models \mathbb{P}\big(\Diamond_{t\leq 2} T_3\big) = 0.50$ and $(AB|T) \models \mathbb{P}\big(\Diamond_{C\leq 6} T_3\big) = 0.50$. Fig. 2 gives a time- and cost-bounded reachability probabilities for $(A|B|T)$ and $(AB|T)$ for a range of bounds. Thus, though the two NPTAs satisfy the same WCTL properties, they are obviously quite different with respect to PWCTL. The NPTA B_r of Fig. 1 is a variant of B, with the uniform delay distribution enforced by the invariant $y \leq 2$ being replaced by an exponential distribution with rate $\frac{1}{2}$. Here $(A|B_r|T)$ satisfies $\mathbb{P}\big(\Diamond_{t\leq 2} T_3\big) \approx 0.41$ and $\mathbb{P}\big(\Diamond_{C\leq 6} T_3\big) \approx 0.49$.

4 Statistical Model Checking for NPTA

As we pointed out, most of model checking problems for NPTAs and PWCTL (including reachability) are undecidable. Our solution is to use a technique that approximates the answer. We rely on *Statistical Model Checking* (SMC) [28,35], that is a series of simulation-based techniques that generate runs of the systems, monitor them, and then use algorithms from statistics to get an estimate of the entire system. At the heart of any SMC approach, there is an algorithm used to generate runs of the system following a stochastic semantics. We propose such an algorithm for NPTAs corresponding to the stochastic semantics proposed in Section 3. Then, we recap existing statistic algorithms, providing the basis for a first SMC algorithm for NPTAs.

Generating Runs of NPTA SMC is used for properties that can be monitored on finite runs. Here, we propose an algorithm that given an NPTA generates a random run up to a cost bound c (with time bounds being a simple case) of an observer clock C. A run of a NPTA is a sequence of alternations of states $\boldsymbol{s_0} \xrightarrow{d_0} \boldsymbol{s'_0} \xrightarrow{o_0} \boldsymbol{s_1} \xrightarrow{d_1} \dots \boldsymbol{s}_n$ obtained by performing delays d_i and emitting outputs o_i. Here we consider a network of NPTAs with states being of the form $(\boldsymbol{\ell}, \nu)$. We construct random runs according to Algorithm 1. We start from an initial state $(\boldsymbol{\ell_0}, \nu_0)$ and repeatedly concatenate random successor states until we reach the bound c for the given observer clock C. Recall that $\nu(C)$ is the value of C in state $(\boldsymbol{\ell}, \nu)$, and the rate of C in location $\boldsymbol{\ell}$ is $R(C)(\boldsymbol{\ell})$. We use the notation $\oplus$ to concatenate runs and $tail(run)$ to access the last state of a run and $delay(\mu_s)$ returns a random delay according to the delay density function μ_s as described in Section 3. The statement "pick" means choose uniformly among the possible choices. Lines 5-6 stop the delay when the runs reach their time bounds with the values of the clocks depending on their rates. The Algorithm 1 may be seen to be correct with respect to the stochastic semantics of NPTAs given in Section 3 in the sense that the probability of the (random) run $RR_{\boldsymbol{\mathcal{A}}}\big((\boldsymbol{\ell_0}, \nu_0), C, c\big)$ satisfying $\Diamond_{C\leq c}\varphi$ is $\mathbb{P}_{\boldsymbol{\mathcal{A}}}\Big(\Diamond_{C\leq c}\, \varphi\Big)$.

Statistical Model Checking Algorithms We briefly recap statistical algorithms permitting to answer the following two types of questions : (1) *Qualitative :* Is the probability for a given NPTA $\boldsymbol{\mathcal{A}}$ to satisfy a property $\Diamond_{C\leq c}\varphi$ greater or equal to a certain threshold θ ? and (2) *Quantitative :* What is the probability for $\boldsymbol{\mathcal{A}}$ to satisfy $\Diamond_{C\leq c}\varphi$. Each run of the system is encoded as a

Algorithm 1: Random run for a NPTA-network $\mathcal{A}$

function $RR_{\mathcal{A}}((\boldsymbol{\ell_0}, \nu_0), C, c)$
1 $run := (\boldsymbol{\ell}, \nu) := tail(run) := (\boldsymbol{\ell_0}, \nu_0)$
2 **while** $\nu(C) < c$ **do**
3 **for** $i = 1$ to $|\boldsymbol{\ell}|$ **do** $d_i := delay(\mu_{(\ell_i, \nu_i)})$
4 $d := min_{1 \leq i \leq |\boldsymbol{\ell}|}(d_i)$
5 **if** $d = +\infty \vee \nu(C) + d * R(\boldsymbol{\ell})(C) \geq c$ **then**
6 $d := (\nu(C) - c)/R(\boldsymbol{\ell})(C)$
7 **return** $run \oplus \xrightarrow{d} (\boldsymbol{\ell}, \nu + d * R(\boldsymbol{\ell}))$
  end
8 **else**
9 pick k such that $d_k = d$; $\nu_d := \nu + d * R(\boldsymbol{\ell})$
10 pick $\ell_k \xrightarrow{g,o,r} \ell'_k$ with $g(\nu_d)$
11 $run := run \oplus \xrightarrow{d} (\boldsymbol{\ell}, \nu_d) \xrightarrow{g,o,r} (\boldsymbol{\ell}[l'_k/l_k], [r \mapsto 0](\nu_d))$
  end
12 $(\boldsymbol{\ell}, \nu) := tail(run)$
end
return run

Bernoulli random variable that is true if the run satisfies the property and false otherwise.

Qualitative Question This problem reduces to test the hypothesis $H : p = \mathbb{P}_{\mathcal{A}}(\Diamond_{C \leq c} \varphi) \geq \theta$ against $K : p < \theta$. To bound the probability of making errors, we use strength parameters α and β and we test the hypothesis $H_0 : p \geq p_0$ and $H_1 : p \leq p_1$ with $p_0 = \theta + \delta_0$ and $p_1 = \theta - \delta_1$. The interval $p_0 - p_1$ defines an indifference region, and p_0 and p_1 are used as thresholds in the algorithm. The parameter α is the probability of accepting H_0 when H_1 holds (false positives) and the parameter β is the probability of accepting H_1 when H_0 holds (false negatives). The above test can be solved by using Wald's sequential hypothesis testing [34]. This testcomputes a proportion r among those runs that satisfy the property. With probability 1, the value of the proportion will eventually cross $\log(\beta/(1-\alpha)$ or $\log((1-\beta)/\alpha)$ and one of the two hypothesis will be selected.

Quantitative Question This algorithm [19] computes the number N of runs needed in order to produce an approximation interval $[p-\epsilon, p+\epsilon]$ for $p = Pr(\psi)$ with a confidence $1 - \alpha$. The values of ϵ and α are chosen by the user and N relies on the Chernoff-Hoeffding bound.

5 Beyond "Classical" Statistical Model-Checking

Here, we want to compare $p_1 = \mathbb{P}_{\mathcal{A}}(\Diamond_{C_1 \leq c_1} \varphi_1)$ and $p_2 = \mathbb{P}_{\mathcal{A}}(\Diamond_{C_2 \leq c_2} \varphi_2)$ without computing them. This comparison has clear practical applications e.g. it can be used to compare the performances of an original program with one of its newly

designed extensions. This comparison cannot be performed with the algorithm presented in the previous section. Moreover, using Monte Carlo to estimate the probabilities (which is costly) would not help as both such probabilities would be estimated with different confidences that could hardly be related[10]. In [34], Wald has shown that this problem can be reduced to a sequential hypothesis testing one. Our contributions here are (1) to apply this algorithm in the formal verification area, (2) to extend the original algorithm of [34] to handle cases where we observe the same outcomes for both experiments, and (3) to implement a parametric extension of the algorithm that allows to reuse results on several timed bounds. More precisely, instead of comparing two probabilities with one common cost bound $C \leq c$, the new extension does it for all the N bounds $i * c/N$ with $i = 1 \dots N$ by reusing existing runs.

Comparison Algorithm. Let the efficiency of satisfying $\Diamond_{C_1 \leq c_1} \varphi_1$ over runs be given by $k_1 = p_1/(1-p_1)$ and similarly for $\Diamond_{C_2 \leq c_2} \varphi_2$. The relative superiority of "φ_2 over φ_1" is measured by the ratio $u = \frac{k_2}{k_1} = \frac{p_2(1-p_1)}{p_1(1-p_2)}$. If $u = 1$ both properties are equally good, if $u > 1$, φ_2 is better, otherwise φ_1 is better. Due to indifference region, we have two parameters u_0 and u_1 such that $u_0 < u_1$ to make the decision. If $u \leq u_0$ we favor φ_1 and if $u \geq u_1$ we favor φ_2. The parameter α is the probability of rejecting φ_1 when $u \leq u_0$ and the parameter β is the probability of rejecting φ_2 when $u \geq u_1$. An outcome for the comparison algorithm is a pair $(x_1, x_2) = (r_1 \models \Diamond_{C_1 \leq c_1} \varphi_1, r_2 \models \Diamond_{C_2 \leq c_2} \varphi_2)$ for two independent runs r_1 and r_2. In Wald's version (lines 10–14 of Algorithm 2), the outcomes $(0,0)$ and $(1,1)$ are ignored. The algorithm works if it is guaranteed to eventually generate different outcomes. We extend the algorithm with a qualitative test (lines 5–9 of Algorithm 2) to handle the case when the outcomes are always the same. The hypothesis we test is $\mathbb{P}_{\boldsymbol{\mathcal{A}}}((r_1 \models \Diamond_{C_1 \leq c_1} \varphi_1) = (r_2 \models \Diamond_{C_2 \leq c_2} \varphi_2)) \geq \theta$ for two independent runs r_1 and r_2. We note that this does not affect the correctness of the original algorithm for accepting or rejecting process 2. The modified algorithm now returns *indifferent* in addition, which corresponds to our added hypothesis to cut down the number of necessary runs[11]. Typically we want the parameters $p'_0 = \theta + \delta_0$ (for the corresponding hypothesis H_0) and $p'_1 = \theta - \delta_1$ (for H_1) to be close to 1. Our version of the comparison algorithm is shown in algorithm 2 with the following initializations:

$$a = \frac{\log(\frac{\beta}{1-\alpha})}{\log(u_1)-\log(u_0)}, r = \frac{\log(\frac{1-\beta}{\alpha})}{\log(u_1)-\log(u_o)}, c = \frac{\log(\frac{1+u_1}{1+u_0})}{\log(u_1)-\log(u_o)}$$

Parametrised Comparisons We now generalise the comparison algorithm to give answers not only for one cost bound c but N cost bounds $i * c/N$ (with $i = 1 \dots N$). This algorithm is of particular interest to generate distribution over timed bounds value of the property. The idea is to reuse the runs of smaller bounds. When $\Diamond_{C \leq c} \varphi_1$ or $\Diamond_{C \leq c} \varphi_2$ holds on some run we keep track of the corresponding point in cost (otherwise the cost value is irrelevant). Every pair or

[10] Interleaving intervals for the estimate (even with same confidence) may give non-deterministic results, not to mention that computing estimates is more expensive than hypothesis testing in terms of runs.

[11] This also frees us from the assumption that the processes have some different outputs.

Algorithm 2: Comparison of probabilities

```
   function comprise(S:model , ψ1, ψ2: properties)
 1 check := 1, q := 0, t := 0
 2 while true do
 3     Observe the random variable x1 corresponding to ψ1 for a run.
 4     Observe the random variable x2 corresponding to ψ2 for a run.
 5     if check = 1 then
 6         x := (x1 == x2)
 7         q := q + x * log(p1'/p0') + (1 − x) * log((1 − p1')/(1 − p0'))
 8         if q ≤ log(β/(1 − α)) then return indifferent
 9         if r ≥ log((1 − β)/α) then check := 0
       end
10     if x1 ≠ x2 then
11         a := a + c, r := r + c
12         if x1 = 0 and x2 = 1 then t := t + 1
13         if t ≤ a then accept process 2.
14         if t ≥ r then reject process 2.
       end
   end
```

runs gives a pair of outcomes (x_1, x_2) at cost points (c_1, c_2). For every $i = 1 \ldots N$ we define the new pair of outcomes $(y_{i_1}, y_{i_2}) = \big(x_1 \wedge (i \cdot c/N \geq t_1 \cdot rate_C), x_2 \wedge (i \cdot c/N \geq t_2 \cdot rate_C)\big)$ for which we use our comparison algorithm. We terminate the algorithm when a result for every i^{th} bound is known.

6 Case Studies

We have extended Uppaal with the algorithms described in this paper. The implementation provides access to all the powerful features of the tool, including user defined functions and types, and use of expressions in guards, invariants, clock-rates as well as delay-rates. Also the implementation supports branching edges with discrete probabilities (using weights), thus supporting probabilistic timed automata (a feature for which our stochastic semantics of NPTA may be easily extended). Besides these additional features, the case-studies reported below (as well as the plots in the previous part of the paper) illustrate the nice features of the new plot composing GUI of the tool[12]. Our objective here is not to study the evolutions of performances with the increase of condidence level, but rather to give a sample of case studies on which our approach can be applied.

Train-Gate Example We consider the train-gate example [5], where N trains want to cross a one-track bridge. We extend the original model by specifying an arrival rate for Train i $((i+1)/N)$. Trains are then approaching, but they can be stopped before some time threshold. When a train is stopped, it can start again.

[12] *http://www.cs.aau.dk/~adavid/smc/* for details.

Eventually trains cross the bridge and go back to their safe state. The template of these trains is given in Fig. 3(a). Our model captures the natural behavior of arrivals with some exponential rate and random delays chosen with uniform distributions in states labelled with invariants. The tool is used to estimate the probability that Train 0 and Train 5 will cross the bridge in less than 100 units of time. Given a confidence level of 0.05 the confidence intervals returned are $[0.541, 0.641]$ and $[0.944, 1]$. The tool computes for each time bound T the frequency count of runs of length T for which the property holds. Figure 3(b) shows a superposition of both distributions obtained directly with our tool that provides a plot composer for this purpose.

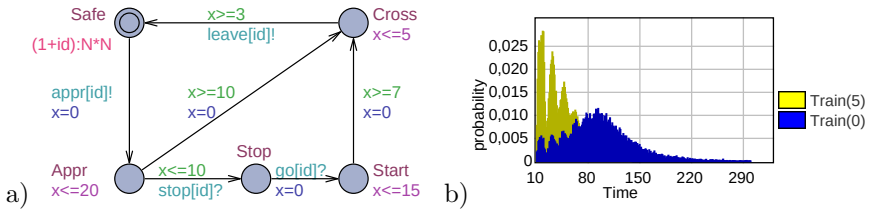

Figure 3: Template of a train (a) and probability density distributions for $\Diamond_{T \leq t}$`Train(0).Cross` and $\Diamond_{T \leq t}$`Train(5).Cross`.

The distribution for Train 5 is the one with higher probability at the beginning, which confirms that this train is indeed the faster one. An interesting point is to note the valleys in the probability densities that correspond to other trains conflicting for crossing the bridge. They are particularly visible for Train 0. The number of valleys corresponds to the number of trains. This is clearly not a trivial distribution (not even uni-modal) that we could not have guessed manually even from such a simple model. In addition, we use the qualitative check to cheaply refine the bounds to $[0.541, 0.59]$ and $[0.97, 1]$.

We then compare the probability for Train 0 to cross when all other trains are stopped with the same probability for Train 5. In the first plot (Fig. 4 top), we check the same property with 100 different time bounds from 10 to 1000 in steps of 10 and we plot the number of runs for each check. These experiments only check for the specified bound, they are not parametrised. In the second plot, we use the parametric extension presented in Section 5 with a granularity of 10 time units. We configured the thresholds u_0 and u_1 to differentiate the comparisons at $u_0 = 1 - \epsilon$ and $u_1 = 1 + \epsilon$ with $\epsilon = 0.1, 0.05, 0.01$ as shown on the figure. In addition, we use a larger time bound to visualise the behaviors after time 600 that are interesting for our checker. In the first plot of Fig. 4, we show for each time bound the average of runs needed by the comparison algorithm repeated 30 times for different values of ϵ. In the bottom plot, we first superpose the cumulative probability for both trains (curves Train 0 and Train 5) that we obtain by applying the quantitative algorithm of Section 4 for each time bound in the sampling. Interestingly, before that point, train 5 is better and later train 0 is better. Second, we compare these probabilities by using the comparison algorithm (curves 0.1 0.05 0.01). This algorithm can retrieve 3

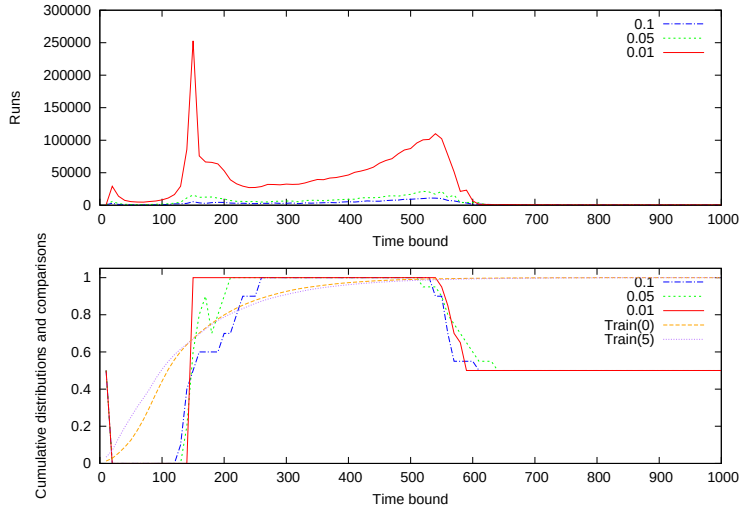

Figure 4: Comparing trains 0 and 5.

values: 0 if Train 0 wins, 1 if Train 5 wins and 0.5 otherwise. We report for each time bound and each value of ϵ the average of these values for 30 executions of the algorithm.

In addition, to evaluate the efficiency of computing all results at once to obtain these curves, we measure the accumulated time to check all the 100 properties for the first plot (sequential check), which takes $92s, 182s, 924s$ for $\epsilon = 0.1, 0.05, 0.01$, and the time to obtain all the results at once (parallel check), which takes $5s, 12s, 92s$. The experiments are done on a Pentium D at 2.4GHz and consume very little memory. The parallel check is about 10 times faster[13]. In fact it is limited by the highest number of runs required as shown by the second peak in Fig. 4. The expensive part is to generate the runs so reusing them is important. Note that at the beginning and at the end, our algorithm aborts the comparison of the curves, which is visible as the number of runs is sharply cut.

Lightweight Media Access Control Protocol (LMAC). This protocol is used in sensor networks to schedule communication between nodes. It is targeted for distributed self-configuration, collision avoidance and energy efficiency. In this study we reproduce the improved UPPAAL model from [15] without verification optimisations, parametrise with network topology (ring and chain), add probabilistic weights (exponential and uniform) over discrete delay decisions and examine statistical properties which were not possible to check before. Based on [33], our node model consumes 21, 22, 2 and 1 power units when a node is sending, receiving, listening for messages or being idle respectively.

Fig. 5a shows that collisions may happen in all cases and the probability of collision is higher with exponential decision weights than uniform decision

[13] The implementation checks simulations sequentially using a single thread.

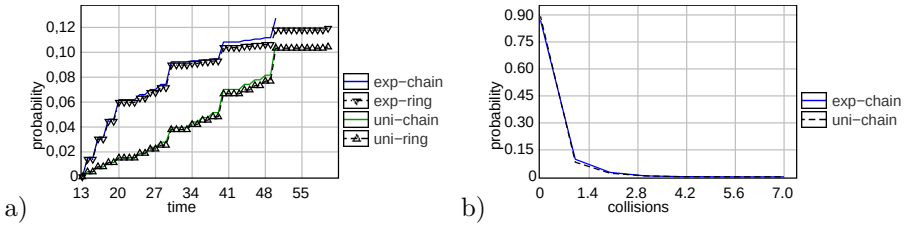

Figure 5: Collision probabilities when using exponential and uniform weights in chain and ring topologies, a) cumulative probability of collision over time and b) probability of having various numbers of collisions.

weights, but seems independent of topology (ring or chain). The probability of collision stays stable after 50 time units, despite longer simulations, meaning that the network may stay collision free if the first collisions are avoided. We also applied the method for parametrised probability comparison for the collision probability. The results show that up to 14 time units the probabilities are the same and later exponential weights have higher collision probability than uniform, but the results were inconclusive when comparing different topologies.

The probable collision counts in the chain topology are shown in Fig. 5b, where the case with 0 collisions has a probability of 87.06% and 89.21% when using exponential and uniform weights respectively. The maximum number of probable collisions is 7 for both weight distributions despite very long runs, meaning that the network eventually recovers from collisions.

Fig. 6 shows energy consumption probability density: using uniform and exponential weights in a chain and a ring topologies. The probability `Pr[energy <= 50000](<> time>=1000)` as estimated. Ring topology uses more power (possibly due to collisions), and uniform weights use slightly less energy than exponential weights in these particular topologies.

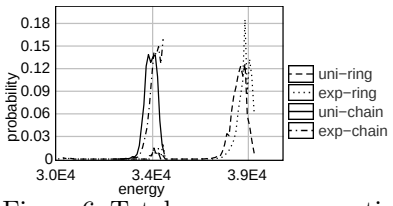

Figure 6: Total energy consumption.

Duration Probabilistic Automata (DPA) [21]. Those automata are used for modelling job-shop problems. A DPA consists of several Simple DPAs (SDPA). An SDPA is a processing unit, a clock and a list of tasks to process sequentially. Each task has an associated duration interval, from which its duration is chosen (uniformly). Resources are used to model task races – we allow different resource types and different quantities of each type. A fixed priority scheduler is used to resolve conflicts. An example is shown in Fig. 7. DPA can be encoded in our tool

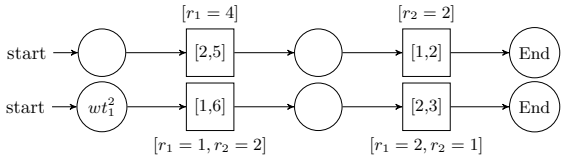

Figure 7: Rectangles are busy states and circles are for waiting when resources are not available. There are $r_1 = 5$ and $r_2 = 3$ resources available.

(continuous or discrete time semantics) or in PRISM (discrete semantics), see the technical report [27]. In PRISM, integer and boolean variables are used to encode the current tasks and resources. PRISM only supports the discrete time model. In UPPAAL, a chain of waiting and task locations is created for each SDPA. Guards and invariants encode the duration of the task, and an array of integers contain the available resources. The scheduler is encoded as a separate template. For UPPAAL, we have modelled a discrete version as close as possible to the PRISM model (Up_p), an improved discrete version that "jumps" to interesting points in time (Up_d), and a continuous time version that making full use of our formalism (Up_c).

Table 1: Performance of SMC (sec)

Param.			**Estim.**				**Hyp. Testing**			
n	k	m	PRISM	Up_p	Up_d	Up_c	PRISM	Up_p	Up_d	Up_c
4	4	3	2.7	0.3	0.2	0.2	2.0	0.1	0.1	0.1
6	6	3	7.7	0.6	0.5	0.4	3.9	0.2	0.2	0.3
8	8	3	26.5	1.2	0.9	0.7	16.4	0.5	0.4	0.3
20	40	20	>300				>300	35.5	26.2	20.7
30	40	20	>300				>300	61.2	41.8	33.2
40	40	20	>300				>300	92.2	56.9	59.5
40	20	20	>300				>300	41.1	31.2	26.5
40	30	20	>300				>300	68.8	46.7	46.1
40	55	40	>300				>300			219.5

The performance of the translations is shown in Tab. 1, based on DPAs with n SDPAs, k tasks per SDPA and m resource types. The resource usage and duration interval are randomised. In the hypothesis testing column, UPPAAL uses the sequential hypothesis testing introduced in Section 4, whereas PRISM uses its own new implementation of the hypothesis testing algorithm. In the estimation column, both UPPAAL and PRISM use the quantitative check of Section 4, but UPPAAL is faster thanks to its more suitable formalism. For both tools, the error bounds used are $\alpha = \beta = 0.05$. In the hypothesis test, the indifference region size is 0.01, while we have $\epsilon = 0.05$ for the quantitative approach. The query for the approximation test is: "What is the probability of all SDPAs ending within t time units?", and for hypothesis testing it is: "Do all SDPAs end within t time units with probability greater than 40%?". The value of t varies for each model as it was computed by simulating the system 369 times and represent the value for which at least 60% of the runs reached the final state. Each number in the table is the average of 10 SMC analyses on the given model. The results show that UPPAAL is an order of magnitude faster than PRISM even with the discrete encoding, which puts UPPAAL at a disadvantage given that it is designed for continuous time[14].

Acknowledgments

We thank Holger Hermans who helped us to clarify the relation between WCTL and the conservative stochastic extension considered in this paper. We are also thankful to David Parker for fruitful discussions on SMC and PRISM.

[14] We note that the number of steps generated for the runs of the PRISM model and the discrete UPPAAL model upp_p are comparable.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
2. R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In Benedetto and Sangiovanni-Vincentelli [7], pages 49–62.
4. Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Probabilistic and topological semantics for timed automata. In *FSTTCS*, volume 4855 of *LNCS*, pages 179–191. Springer, 2007.
5. Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on Uppaal. In Marco Bernardo and Flavio Corradini, editors, *SFM*, ncs(3185), pages 200–236. Springer, 2004.
6. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In Benedetto and Sangiovanni-Vincentelli [7], pages 147–161.
7. Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors. *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Proceedings*, volume 2034 of *LNCS*. Springer, 2001.
8. Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *QEST*, pages 55–64. IEEE Computer Society, 2008.
9. J. Bogdoll, L-M Fiorti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FORTE*, LNCS. Springer, 2011. to appear.
10. H. Bohnenkamp, P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. Modest: A compositional modeling formalism for real-time and stochastic systems. Technical Report CTIT 04-46, University of Twente, 2004.
11. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
12. E. M. Clarke, A. Donzé, and A. Legay. Statistical model checking of mixed-analog circuits with an application to a third order delta-sigma modulator. In *HVC*, volume 5394 of *LNCS*, pages 149–163. Springer, 2008.
13. A. David, K.G. Larsen, A. Legay, Z.Wang, and M. Mikucionis. Time for real statistical model-checking: Statistical model-checking for real-time systems. In *CAV*, LNCS. Springer, 2011.
14. Alexandre David, Kim.G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Timed I/O automata: a complete specification theory for real-time systems. In *HSCC*. ACM, 2010.
15. Ansgar Fehnker, Lodewijk van Hoesel, and Angelika Mader. Modelling and verification of the lmac protocol for wireless sensor networks. In Jim Davies and Jeremy Gibbons, editors, *Integrated Formal Methods*, volume 4591 of *LNCS*, pages 253–272. Springer Berlin / Heidelberg, 2007.
16. Goran Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *STTT*, 10(3):263–279, 2008.
17. Rodolfo Gómez. A compositional translation of timed automata with deadlines to uppaal timed automata. In *Formal Modeling and Analysis of Timed Systems*, volume 5813, pages 179–194, 2009.

18. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, LNCS, pages 73–84, 2004.
19. Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In *VMCAI*, volume 2937 of *LNCS*, pages 307–329. Springer, 2003.
20. Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A bayesian approach to model checking biological systems. In *CMSB*, volume 5688 of *LNCS*, pages 218–234. Springer, 2009.
21. V. Kaczmarczyk, M. Sir, and Z. Bradac. Stochastic timed automata simulator. In *4th European Computing Conference*. WSEAS.
22. Vaclav Kaczmarczyk, Michal Sir, and Zdenek Bradac. Stochastic timed automata simulator. 2010. In Proceedings of the 4th European Computing Conference.
23. Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. In *POPL*, pages 344–352, 1989.
24. Oded Maler, Kim G. Larsen, and Bruce H. Krogh. On zone-based analysis of duration probabilistic automata. In *INFINITY*, volume 39 of *EPTCS*, pages 33–46, 2010.
25. M. Minea. *Partial Order Reduction for Verification of Timed Systems*. PhD thesis, Carnegie Mellon, 1999.
26. P. Panangaden. *Labelled Markov Processes*. Imperial College Press, 2010.
27. D. Poulsen and J. van Vliet. Duration probabilistic automata. Technical report, Aalborg University, 2011.
28. Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.
29. The smv model checker. Available at `http://www.kenmcmil.com/smv.html` .
30. The spin tool (spin). Available at `http://spinroot.com/spin/whatispin.html`.
31. Tino Teige, Andreas Eggers, and Martin Fränzle. Constraint-based analysis of concurrent probabilistic hybrid systems: An application to networked automation systems. *Nonlinear Analysis: Hybrid Systems*, 2011.
32. The uppaal tool. Available at `http://www.uppaal.com/`.
33. Lodewijk Frans Willem van Hoesel. *Sensors on speaking terms: schedule-based medium access control protocols for wireless sensor networks*. PhD thesis, University of Twente, June 2007.
34. R. Wald. *Sequential Analysis*. Dove Publisher, 2004.
35. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.
36. Håkan L. S. Younes. Ymer: A statistical model checker. In *CAV*, volume 3576 of *LNCS*, pages 429–433. Springer, 2005.
37. Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, LNCS 2404, pages 223–235. Springer, 2002.
38. P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *HSCC*, pages 243–252. ACM ACM, 2010.